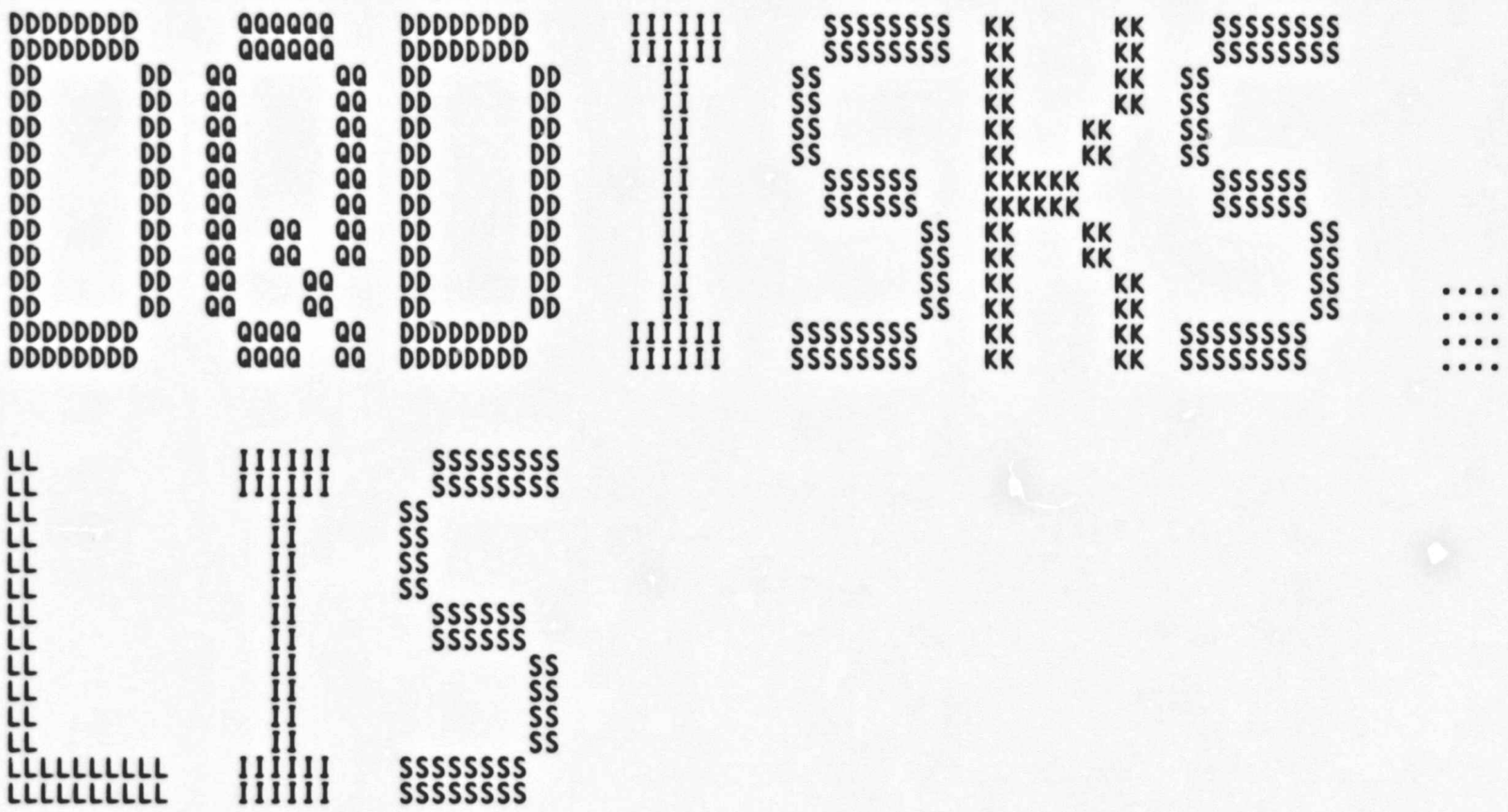
```
EEEEEEEEEEEEEEE   RRRRRRRRRRR      FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE   RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEE               RRR          RRR  FFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEEEEEEEEEEE      RRRRRRRRRRRR      FFFFFFFFFFFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR    RRR        FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEE               RRR       RRR     FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
EEEEEEEEEEEEEEE   RRR          RRR   FFF
```

FILEID**DQDISKS



```
0001    C
0002    C Version:      'V04-000'
0003    C
0004    C****************************************************************************
0005    C*                                                                          *
0006    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0007    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0008    C*  ALL RIGHTS RESERVED.                                                    *
0009    C*                                                                          *
0010    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0011    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0012    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0013    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0014    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0015    C*  TRANSFERRED.                                                            *
0016    C*                                                                          *
0017    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0018    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0019    C*  CORPORATION.                                                            *
0020    C*                                                                          *
0021    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0022    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0023    C*                                                                          *
0024    C*                                                                          *
0025    C****************************************************************************
0026    C
0027
0028    c       Author Brian Porter                     Creation Date   20-JUL-1981
0029
0030    c++
0031    c       Functional description:
0032    c
0033    c       This module is used to display error log entries logged for the
0034    c       11/7zz IDC.  The format of the device specific portion of the record
0035    c       is as follows.
0036    c
0037    c       +-----------------------------------+
0038    c       |               csr                 |
0039    c       +-----------------------------------+
0040    c       |               bar                 |
0041    c       +-----------------------------------+
0042    c       |               bcr                 |
0043    c       +-----------------------------------+
0044    c       |               dar                 |
0045    c       +-----------------------------------+
0046    c       |               mpr                 |
0047    c       +-----------------------------------+
0048    c       |               ecc1                |
0049    c       +-----------------------------------+
0050    c       |               ecc2                |
0051    c       +-----------------------------------+
0052    c       |        data path number           |
0053    c       +-----------------------------------+
0054    c       |     data path reg (always 0)      |
0055    c       +-----------------------------------+
0056    c       |           final uba map           |
0057    c       +-----------------------------------+
```

```
0058     c        ¦        previous uba map          ¦
0059     c        +----------------------------------+
0060     c        ¦     vec$l_mapreg (from crb)      ¦
0061     c        +----------------------------------+
0062     c
0063     c        Modified by:
0064     c
0065     c        V03-003 SAR0217         Sharon A. Reynolds,     28-Mar-1984
0066     c                Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
0067     c
0068     c        V03-002 SAR0069         Sharon A. Reynolds,     20-Jun-1983
0069     c                Changed the carriage control in the 'format' statements
0070     c                for use with ERF.
0071     c
0072     c        v03-001 SAR0046         Sharon A. Reynolds,     9-Jun-1983
0073     c                Removed brief/cryptic support.
0074     c
0075     c        v02-008 BP0008          Brian Porter,           23-JAN-1982
0076     c                Corrected polarity of 'plug valid' for the r80.
0077     c
0078     c        v02-007 BP0007          Brian Porter,           23-NOV-1981
0079     c                Minor edit.
0080     c
0081     c        v02-006 BP0006          Brian Porter,           04-NOV-1981
0082     c                Corrected 'DAR' output error.  Added 'device attention'
0083     c                support.
0084     c
0085     c        v02-005 BP0005          Brian Porter,           30-SEP-1981
0086     c                Corrected random problems.
0087     c
0088     c        v02-004 BP0004          Brian Porter,           29-SEP-1981
0089     c                Added 'DAR' decoding functionality.
0090     c
0091     c        v02-003 BP0003          Brian Porter,           14-SEP-1981
0092     c                Corrected problem in attention logic.  Added CSR
0093     c                functionality.
0094     c
0095     c        v02-002 BP0002          Brian Porter,           31-AUG-1981
0096     c                Corrected call to calc_map.
0097     c
0098     c        v02-001 BP0001          Brian Porter,           24-AUG-1981
0099     c                Changed record format to conform to other drivers.
0100     c**
0101     c--
0102
0103
0104
0105              Subroutine DQDISKS (lun)
0106
0107
0108              include 'src$:msghdr.for /nolist'
0167              include 'src$:deverr.for /nolist'
0268
0269              byte            lun
0270
0271              integer*4       control_status_register
0272              integer*4       bus_address_register
```



```
032
032
032
032
032
032
032
032
032
033
033
033
033
033
033
033
033
033
034
034
034
034
034
034
034
034
034
035
035
035
035
035
```

```
0273            integer*4       byte_control_register
0274            integer*4       disk_address_register
0275            integer*4       multi_purpose_register
0276            integer*4       ecc_position_register
0277            integer*4       ecc_pattern_register
0278            integer*4       data_path_number
0279            integer*4       data_path_register
0280            integer*4       final_map_register
0281            integer*4       previous_map_register
0282            integer*4       vec$l_mapreg
0283
0284            equivalence     (emb$l_dv_regsav(0),control_status_register)
0285            equivalence     (emb$l_dv_regsav(1),bus_address_register)
0286            equivalence     (emb$l_dv_regsav(2),byte_control_register)
0287            equivalence     (emb$l_dv_regsav(3),disk_address_register)
0288            equivalence     (emb$l_dv_regsav(4),multi_purpose_register)
0289            equivalence     (emb$l_dv_regsav(5),ecc_position_register)
0290            equivalence     (emb$l_dv_regsav(6),ecc_pattern_register)
0291            equivalence     (emb$l_dv_regsav(7),data_path_number)
0292            equivalence     (emb$l_dv_regsav(8),data_path_register)
0293            equivalence     (emb$l_dv_regsav(9),final_map_register)
0294            equivalence     (emb$l_dv_regsav(10),previous_map_register)
0295            equivalence     (emb$l_dv_regsav(11),vec$l_mapreg)
0296
0297            character*12    v1csr(0:0)
0298            data            v1csr(0)        /'DRIVE READY*'/
0299
0300            character*17    v2csr(6:7)
0301            data            v2csr(6)        /'INTERRUPT ENABLE*'/
0302            data            v2csr(7)        /'CONTROLLER READY*'/
0303
0304            character*21    v3csr(10:10)
0305            data            v3csr(10)       /'OPERATION INCOMPLETE*'/
0306
0307            character*20    v4csr(13:15)
0308            data            v4csr(13)       /'NON-EXISTENT MEMORY*'/
0309            data            v4csr(14)       /'DRIVE ERROR*'/
0310            data            v4csr(15)       /'COMPOSITE ERROR*'/
0311
0312            character*22    v5csr(22:24)
0313            data            v5csr(22)       /'R80 SKIP SECTOR ERROR*'/
0314            data            v5csr(23)       /'R80 SKIP SECTOR ERROR*'/
0315            data            v5csr(24)       /'INTERRUPT REQUEST*'/
0316
0317            character*30    v6csr(26:28)
0318            data            v6csr(26)       /'R80*'/
0319            data            v6csr(27)       /'AUTOMATIC SKIP SECTOR INHIBIT*'/
0320            data            v6csr(28)       /'TIMEOUT INHIBIT*'/
0321
0322            character*11    v1rl02_mpr(3:5)
0323            data            v1rl02_mpr(3)   /'BRUSH HOME*'/
0324            data            v1rl02_mpr(4)   /'HEADS OUT*'/
0325            data            v1rl02_mpr(5)   /'COVER OPEN*'/
0326
0327            character*19    v2rl02_mpr(8:15)
0328            data            v2rl02_mpr(8)   /'DRIVE SELECT ERROR*'/
0329            data            v2rl02_mpr(9)   /'VOLUME CHECK*'/
```

```
0330            data            v2rl02_mpr(10)  /'WRITE GATE ERROR*'/
0331            data            v2rl02_mpr(11)  /'SPINDLE ERROR*'/
0332            data            v2rl02_mpr(12)  /'SEEK TIMEOUT*'/
0333            data            v2rl02_mpr(13)  /'WRITE LOCK*'/
0334            data            v2rl02_mpr(14)  /'HEAD CURRENT ERROR*'/
0335            data            v2rl02_mpr(15)  /'WRITE DATE ERROR*'/
0336
0337            character*14    v1r80_mpr(8:13)
0338            data            v1r80_mpr(8)    /'FAULT*'/
0339            data            v1r80_mpr(9)    /'PLUG VALID*'/
0340            data            v1r80_mpr(10)   /'SEEK ERROR*'/
0341            data            v1r80_mpr(11)   /'ON CYLINDER*'/
0342            data            v1r80_mpr(12)   /'DRIVE READY*'/
0343            data            v1r80_mpr(13)   /'WRITE PROTECT*'/
0344
0345            integer*4       compress4
0346            integer*4       compressc
0347            integer*4       field
0348
0349            character*27    idc_command(0:7)
0350            data            idc_command(0)  /'NO DRIVE OPERATION*'/
0351            data            idc_command(1)  /'WRITE CHECK DATA*'/
0352            data            idc_command(2)  /'GET STATUS*'/
0353            data            idc_command(3)  /'SEEK*'/
0354            data            idc_command(4)  /'READ HEADER*'/
0355            data            idc_command(5)  /'WRITE DATA*'/
0356            data            idc_command(6)  /'READ DATA*'/
0357            data            idc_command(7)  /'READ DATA W/O HEADER CHECK*'/
0358
0359            logical*1       diagnostic_mode
0360
0361            integer*4       lib$extzv
0362            integer*4       data_check_and_opi_bits
0363            integer*4       data_late_and_opi_bits
0364            integer*4       sector_count
0365            integer*4       ecc_status_bits
0366            integer*4       rl02_status_bits
0367
0368            character*20    v1rl02_status_bits(0:7)
0369            data    v1rl02_status_bits(0)   /'LOAD STATE*'/
0370            data    v1rl02_status_bits(1)   /'SPIN UP*'/
0371            data    v1rl02_status_bits(2)   /'BRUSH CYCLE*'/
0372            data    v1rl02_status_bits(3)   /'LOAD HEADS*'/
0373            data    v1rl02_status_bits(4)   /'SEEK TRACK COUNTING*'/
0374            data    v1rl02_status_bits(5)   /'SEEK LINEAR MODE*'/
0375            data    v1rl02_status_bits(6)   /'UNLOAD HEADS*'/
0376            data    v1rl02_status_bits(7)   /'SPIN DOWN*'/
0377
0378            integer*4       device_function
0379            integer*4       device_type
0380            integer*4       sector
0381            integer*4       cylinder
0382            integer*4       tag
0383            integer*4       head
0384
0385            character*11    v1dar(0:1)
0386            data            v1dar(0)        /'MARKER*'/
```

```
0387            data            v1dar(1)        /'GET STATUS*'/
0388
0389            character*6     v2dar(3:3)
0390            data            v2dar(3)        /'RESET*'/
0391
0392            character*8     v4dar(2:2,0:1)
0393            data            v4dar(2,0)      /'REVERSE*'/
0394            data            v4dar(2,1)      /'FORWARD*'/
0395
0396            character*18    v6dar(4:4,0:1)
0397            data            v6dar(4,0)      /'SELECT LOWER HEAD*'/
0398            data            v6dar(4,1)      /'SELECT UPPER HEAD*'/
0399
0400            character*15    v7dar(6:6)
0401            data            v7dar(6)        /'RETURN-TO-ZERO*'/
0402
0403
0404            call frctof (lun)
0405
0406            call dhead1 (lun,'RB730')
0407
0408            diagnostic_mode = .false.
0409
0410            if (lib$extzv(25,1,control_status_register) .eq. 1)
0411            1 diagnostic_mode = .true.
0412
0413            device_function = lib$extzv (1,3,control_status_register)
0414
0415            device_type = lib$extzv (26,1,control_status_register)
0416
0417            call linchk (lun,2)
0418
0419            write(lun,5) 'RB CSR',control_status_register
0420    5       format(/' ',t8,a,t24,z8.8)
0421
0422            if (.not. diagnostic_mode) then
0423
0424            call output (lun,control_status_register,v1csr,0,0,0,'0')
0425
0426            call linchk (lun,1)
0427
0428            if (lib$extzv(29,1,control_status_register) .eq. 1) then
0429
0430            write(lun,10) 'R80 WRITE FORMAT FUNCTION'
0431    10      format(' ',t40,a)
0432            else
0433
0434            idc_function = lib$extzv(1,3,control_status_register)
0435
0436            write(lun,15) idc_command(idc_function)
0437    15      format(' ',t40,a<compressc (idc_command(idc_function))>)
0438            endif
0439
0440            call output (lun,control_status_register,v2csr,6,6,7,'0')
0441
0442            call linchk (lun,1)
0443
```

```
0444            write(lun,20) 'DRIVE #',lib$extzv(8,2,control_status_register),
0445            1 '. SELECTED'
0446      20    format(' ',t40,a,i1.1,a)
0447
0448            call output (lun,control_status_register,v3csr,10,10,10,'0')
0449
0450            data_check_and_opi_bits = lib$extzv(10,2,control_status_register)
0451
0452            if (
0453            1 data_check_and_opi_bits .eq. 2
0454            1 .or.
0455            1 data_check_and_opi_bits .eq. 3
0456            1 ) then
0457
0458            call linchk (lun,1)
0459            endif
0460
0461            if (data_check_and_opi_bits .eq. 2) then
0462
0463            write(lun,25) 'DATA CHECK ERROR'
0464      25    format(' ',t40,a)
0465
0466            else if (data_check_and_opi_bits .eq. 3) then
0467
0468            write(lun,25) 'HEADER CRC ERROR'
0469            endif
0470
0471            data_late_and_opi_bits = lib$extzv(10,3,control_status_register)
0472
0473            if (
0474            1 data_late_and_opi_bits .eq. 4
0475            1 .or.
0476            1 data_late_and_opi_bits .eq. 5
0477            1 ) then
0478
0479            call linchk (lun,1)
0480            endif
0481
0482            if (data_late_and_opi_bits .eq. 4) then
0483
0484            write(lun,25) 'DATA LATE'
0485
0486            else if (data_late_and_opi_bits .eq. 5) then
0487
0488            write(lun,25) 'HEADER NOT FOUND'
0489            endif
0490
0491            call output (lun,control_status_register,v4csr,13,13,15,'0')
0492
0493            do 35,i = 16,19
0494
0495            if (lib$extzv(i,1,control_status_register) .eq. 1) then
0496
0497            call linchk (lun,1)
0498
0499            write(lun,30) 'ATTENTION DRIVE #',i-16,'.'
0500      30    format(' ',t40,a,i1.1,a)
```

```
0501            endif
0502
0503    35      continue
0504
0505            if (lib$extzv (26,1,control_status_register) .eq. 1) then
0506
0507            ecc_status_bits = lib$extzv (20,2,control_status_register)
0508
0509            if (ecc_status_bits .ne. 0) then
0510
0511            call linchk (lun,1)
0512
0513            if (ecc_status_bits .eq. 1) then
0514
0515            write(lun,40) 'DATA ERROR'
0516    40      format(' ',t40,a)
0517
0518            else if (ecc_status_bits .eq. 2) then
0519
0520            write(lun,40) 'HARD ERROR'
0521
0522            else if (ecc_status_bits .eq. 3) then
0523
0524            write(lun,40) 'CORRECTABLE ERROR'
0525            endif
0526            endif
0527            endif
0528
0529            call output (lun,control_status_register,v5csr,22,22,24,'0')
0530
0531            if (lib$extzv (26,1,control_status_register) .eq. 1) then
0532
0533            call output (lun,control_status_register,v6csr,26,26,28,'0')
0534            endif
0535            else
0536
0537            call linchk (lun,1)
0538
0539            write(lun,40) 'DIAGNOSTIC MODE'
0540            endif
0541
0542            call linchk (lun,1)
0543
0544            write(lun,45) 'RB_BAR',bus_address_register
0545    45      format(' ',t8,a,t24,z8.8)
0546
0547            if (.not. diagnostic_mode) then
0548
0549            if (
0550            1 device_function .eq. 1
0551            1 .or.
0552            1 device_function .eq. 5
0553            1 .or.
0554            1 device_function .eq. 6
0555            1 .or.
0556            1 device_function .eq. 7
0557            1 ) then
```

```
0558
0559            call calc_map (lun,16,bus_address_register,bus_address_register)
0560            endif
0561            endif
0562
0563            call linchk (lun,1)
0564
0565            write(lun,45) 'RB BCR',byte_control_register
0566
0567            call linchk (lun,1)
0568
0569            write(lun,45) 'RB DAR',disk_address_register
0570
0571            if (.not. diagnostic_mode) then
0572
0573            if (
0574            1 device_function .eq. 1
0575            1 .or.
0576            1 device_function .eq. 5
0577            1 .or.
0578            1 device_function .eq. 6
0579            1 .or.
0580            1 device_function .eq. 7
0581            1 ) then
0582
0583            if (device_type .eq. 0) then
0584
0585            sector = lib$extzv (0,6,disk_address_register)
0586
0587            cylinder = lib$extzv (7,9,disk_address_register)
0588
0589            else if (device_type .eq. 1) then
0590
0591            sector = lib$extzv (0,5,disk_address_register)
0592
0593            cylinder = lib$extzv (9,10,disk_address_register)
0594            endif
0595
0596            call linchk (lun,2)
0597
0598            write(lun,46) sector,cylinder
0599     46     format(' ',t40,'SECTOR #',i<compress4 (sector)>,'.',/,
0600            1 t40,'CYLINDER #',i<compress4 (cylinder)>,'.')
0601
0602            else if (device_function .eq. 2) then
0603
0604            if (device_type .eq. 0) then
0605
0606            call output (lun,disk_address_register,v1dar,0,0,1,'0')
0607
0608            call output (lun,disk_address_register,v2dar,3,3,3,'0')
0609            endif
0610
0611            else if (device_function .eq. 3) then
0612
0613            if (device_type .eq. 0) then
0614
```

```
0615            call output (lun,disk_address_register,v1dar,0,0,1,'0')
0616
0617            call output (lun,disk_address_register,v4dar,2,2,2,'2')
0618
0619            call output (lun,disk_address_register,v2dar,3,3,3,'0')
0620
0621            call output (lun,disk_address_register,v6dar,4,4,4,'2')
0622
0623            cylinder = lib$extzv (7,9,disk_address_register)
0624
0625            call linchk (lun,1)
0626
0627            write(lun,47) cylinder
0628    47      format(' ',t40,i<compress4 (cylinder)>,'. CYLINDER(S) TO MOVE')
0629
0630            else if (device_type .eq. 1) then
0631
0632            tag = lib$extzv (13,3,disk_address_register)
0633
0634            call linchk (lun,1)
0635
0636            if (tag .eq. 1) then
0637
0638            cylinder = lib$extzv (0,10,disk_address_register)
0639
0640            write(lun,48) 'CYLINDER #',cylinder
0641    48      format(' ',t40,a,i<compress4 (cylinder)>,'. SELECTED')
0642
0643            else if (tag .eq. 2) then
0644
0645            head = lib$extzv (0,4,disk_address_register)
0646
0647            write(lun,48) 'HEAD #',head
0648
0649            else if (tag .eq. 4) then
0650
0651            call output (lun,disk_address_register,v7dar,6,6,6,'0')
0652            endif
0653            endif
0654            endif
0655            endif
0656
0657            call linchk (lun,1)
0658
0659            write(lun,50) 'RB MPR',multi_purpose_register
0660    50      format(' ',t8,a,t24,z8.8)
0661
0662            if (.not. diagnostic_mode) then
0663
0664            if (lib$extzv (26,1,control_status_register) .eq. 1) then
0665
0666            sector_count = lib$extzv (0,5,multi_purpose_register)
0667
0668            call linchk (lun,1)
0669
0670            write(lun,55) 'SECTOR COUNT ',sector_count,'.'
0671    55      format(' ',t40,a,i<compress4 (sector_count)>,a)
```


027
027
027
027
027
027
027
028
028
028
028
028
028
028
028
028
029
029
029
029
029
029
029
029
029
029
030
030
030
030
030
030
030
031
031
031
031
031
031
031
031
031
031
032
032
032
032
032
032
032
032
032
032

```
0672
0673            call output (lun,multi_purpose_register,v1r80_mpr,8,8,13,'0')
0674            else
0675
0676            rl02_status_bits = lib$extzv (0,3,multi_purpose_register)
0677
0678            call linchk (lun,1)
0679
0680            write(lun,60) v1rl02_status_bits(rl02_status_bits)
0681    60      format(' ',t40,a<compressc (v1rl02_status_bits(rl02_status_bits))>)
0682
0683            call output (lun,multi_purpose_register,v1rl02_mpr,3,3,5,'0')
0684
0685            call linchk (lun,1)
0686
0687            if (lib$extzv (6,1,multi_purpose_register) .eq. 1) then
0688
0689            write(lun,65) 'LOWER HEAD SELECTED'
0690    65      format(' ',t40,a)
0691            else
0692
0693            write(lun,65) 'UPPER HEAD SELECTED'
0694            endif
0695
0696            call output (lun,multi_purpose_register,v2rl02_mpr,8,8,15,'0')
0697            endif
0698            endif
0699
0700            call linchk (lun,2)
0701
0702            write(lun,70) 'RB ECC1',ecc_position_register,
0703            1 'RB ECC2',ecc_pattern_register
0704    70      format(' ',t8,a,t24,z8.8,/,t8,a,t24,z8.8)
0705
0706            if (
0707            1 (device_function .eq. 1
0708            1 .or.
0709            1 device_function .eq. 5
0710            1 .or.
0711            1 device_function .eq. 6
0712            1 .or.
0713            1 device_function .eq. 7)
0714            1 .and.
0715            1 emb$w_hd_entry .ne. 98
0716            1 ) then
0717
0718            call uba_datapath (lun,iand(data_path_number,'0000007f'x),
0719            1 data_path_register)
0720
0721            call calc_map2 (16,bus_address_register,bus_address_register,field)
0722
0723            call uba_mapping (lun,field,final_map_register)
0724
0725            if (
0726            1 lib$extzv (16,16,emb$l_dv_iosb1) .gt. 512
0727            1 .and.
0728            1 field .ne. 0
```

```
0729            1 ) then
0730
0731            call uba_mapping (lun,(field-1),previous_map_register)
0732            endif
0733
0734            call vecmapreg (lun,vec$l_mapreg)
0735            endif
0736
0737            call linchk (lun,1)
0738
0739            write(lun,75)
0740    75      format(' ',:)
0741
0742            if (emb$w_hd_entry .ne. 98) then
0743
0744            call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)
0745
0746            call ucb$b_ertmax (lun,emb$b_dv_ertmax)
0747            endif
0748
0749            call orb$l_owner (lun,emb$l_dv_ownuic)
0750
0751            call ucb$l_char (lun,emb$l_dv_char)
0752
0753            call ucb$w_sts (lun,emb$w_dv_sts)
0754
0755            call ucb$l_opcnt (lun,emb$l_dv_opcnt)
0756
0757            call ucb$w_errcnt (lun,emb$w_dv_errcnt)
0758
0759            if (emb$w_hd_entry .ne. 98) then
0760
0761            call ucb$l_media (lun,emb$l_dv_media)
0762
0763            call linchk (lun,1)
0764
0765            write(lun,75)
0766
0767            call dqdisks_qio (lun,emb$w_dv_func)
0768
0769            call irp$w_bcnt (lun,emb$w_dv_bcnt)
0770
0771            call irp$w_boff (lun,emb$w_dv_boff)
0772
0773            call irp$l_pid (lun,emb$l_dv_rqpid)
0774
0775            call irp$q_iosb (lun,emb$l_dv_iosb1)
0776            endif
0777
0778            return
0779            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	2861	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	634	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	2972	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	6979	

ENTRY POINTS

Address	Type	Name
0-00000000		DQDISKS

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000056	I*4	BUS_ADDRESS_REGISTER	3-0000005A	I*4	BYTE_CONTROL_REGISTER
3-00000052	I*4	CONTROL_STATUS_REGISTER	2-00000424	I*4	CYLINDER
2-00000404	I*4	DATA_CHECK_AND_OPI_BITS	2-00000408	I*4	DATA_LATE_AND_OPI_BITS
3-0000006E	I*4	DATA_PATH_NUMBER	3-00000072	I*4	DATA_PATH_REGISTER
2-00000418	I*4	DEVICE_FUNCTION	2-0000041C	I*4	DEVICE_TYPE
2-000003FF	L*1	DIAGNOSTIC_MODE	3-0000005E	I*4	DISK_ADDRESS_REGISTER
3-0000006A	I*4	ECC_PATTERN_REGISTER	3-00000066	I*4	ECC_POSITION_REGISTER
2-00000410	I*4	ECC_STATUS_BITS	3-0000001C	L*1	EMB$B_DV_CLASS
3-00000010	L*1	EMB$B_DV_ERTCNT	3-00000011	L*1	EMB$B_DV_ERTMAX
3-0000003E	L*1	EMB$B_DV_NAMLNG	3-0000003A	L*1	EMB$B_DV_SLAVE
3-0000001D	L*1	EMB$B_DV_TYPE	3-00000036	I*4	EMB$L_DV_CHAR
3-00000012	I*4	EMB$L_DV_IOSB1	3-00000016	I*4	EMB$L_DV_IOSB2
3-00000026	I*4	EMB$L_DV_MEDIA	3-0000004E	I*4	EMB$L_DV_NUMREG
3-0000002E	I*4	EMB$L_DV_OPCNT	3-00000032	I*4	EMB$L_DV_OWNUIC
3-0000001E	I*4	EMB$L_DV_RQPID	3-00000000	I*4	EMB$L_HD_SID
3-0000003F	CHAR	EMB$T_DV_NAME	3-00000024	I*2	EMB$W_DV_BCNT
3-00000022	I*2	EMB$W_DV_BOFF	3-0000002C	I*2	EMB$W_DV_ERRCNT
3-0000003C	I*2	EMB$W_DV_FUNC	3-0000001A	I*2	EMB$W_DV_STS
3-0000002A	I*2	EMB$W_DV_UNIT	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	2-00000400	I*4	FIELD
3-00000076	I*4	FINAL_MAP_REGISTER	2-0000042C	I*4	HEAD
2-00000434	I*4	I	2-00000430	I*4	IDC_FUNCTION
AP-0000004@	L*1	LUN	3-00000062	I*4	MULTI_PURPOSE_REGISTER
3-0000007A	I*4	PREVIOUS_MAP_REGISTER	2-00000414	I*4	RL02_STATUS_BITS
2-00000420	I*4	SECTOR	2-0000040C	I*4	SECTOR_COUNT
2-00000428	I*4	TAG	3-0000007E	I*4	VEC$L_MAPREG

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB$L_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
2-00000228	CHAR	IDC_COMMAND	216	(0:7)
2-00000000	CHAR	V1CSR	12	(0:0)
2-000003A0	CHAR	V1DAR	22	(0:1)
2-000001D4	CHAR	V1R80_MPR	84	(8:13)
2-0000011B	CHAR	V1RL02_MPR	33	(3:5)
2-00000300	CHAR	V1RL02_STATUS_BITS	160	(0:7)
2-0000000C	CHAR	V2CSR	34	(6:7)
2-000003B6	CHAR	V2DAR	6	(3:3)
2-0000013C	CHAR	V2RL02_MPR	152	(8:15)
2-0000002E	CHAR	V3CSR	21	(10:10)
2-00000043	CHAR	V4CSR	60	(13:15)
2-000003BC	CHAR	V4DAR	16	(2:2, 0:1)
2-0000007F	CHAR	V5CSR	66	(22:24)
2-000000C1	CHAR	V6CSR	90	(26:28)
2-000003CC	CHAR	V6DAR	36	(4:4, 0:1)
2-000003F0	CHAR	V7DAR	15	(6:6)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000176	5'	1-00000183	10'	1-0000018A	15'	1-00000196	20'	1-000001A1	25'	1-000001A8	30'
**	35	1-000001B3	40'	1-000001BA	45'	1-000001C6	46'	1-000001F7	47'	1-0000021A	48'
1-00000233	50'	1-0000023F	55'	1-0000024D	60'	1-00000259	65'	1-00000260	70'	1-00000275	75'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
	CALC_MAP		CALC_MAP2	I*4	COMPRESS4	I*4	COMPRESSC		DHEAD1		DQDISKS_QIO
	FRCTOF		IRP$L_PID		IRP$Q_IOSB		IRP$W_BCNT		IRP$W_BOFF	I*4	LIB$EXTZV
	LINCHK		ORB$L_OWNER		OUTPUT		UBA_DATAPATH		UBA_MAPPING		UCB$B_ERTCNT
	UCB$B_ERTMAX		UCB$L_CHAR		UCB$L_MEDIA		UCB$L_OPCNT		UCB$W_ERRCNT		UCB$W_STS
	VECMAPREG										

```
0001
0002
0003
0004
0005          Subroutine DQDISKS_QIO (lun,emb$w_dv_func)
0006
0007
0008
0009
0010          include 'src$:qiocommon.for /nolist'
0274
0275
0276
0277
0278          byte            lun
0279
0280          integer*2       emb$w_dv_func
0281
0282          integer*4       qiocode(0:1,0:63)
0283
0284
0285
0286
0287          if (qiocode(0,0) .eq. 0) then
0288
0289          qiocode(1,00) = %loc(io$_nop)
0290
0291          qiocode(1,02) = %loc(io$_seek)
0292
0293          qiocode(1,03) = %loc(io$_recal)
0294
0295          qiocode(1,04) = %loc(io$_drvclr)
0296
0297          qiocode(1,08) = %loc(io$_packack)
0298
0299          qiocode(1,10) = %loc(io$_writecheck)
0300
0301          qiocode(1,11) = %loc(io$_writepblk)
0302
0303          qiocode(1,12) = %loc(io$_readpblk)
0304
0305          qiocode(1,14) = %loc(io$_readhead)
0306
0307          qiocode(1,26) = %loc(io$_setchar)
0308
0309          qiocode(1,27) = %loc(io$_sensechar)
0310
0311          qiocode(1,32) = %loc(io$_writelblk)
0312
0313          qiocode(1,33) = %loc(io$_readlblk)
0314
0315          qiocode(1,35) = %loc(io$_setmode)
0316
0317          qiocode(1,39) = %loc(io$_sensemode)
0318
0319          qiocode(1,48) = %loc(io$_writevblk)
0320
```

```
0321            qiocode(1,49) = %loc(io$_readvblk)
0322
0323            qiocode(1,50) = %loc(io$_access)
0324
0325            qiocode(1,51) = %loc(io$_create)
0326
0327            qiocode(1,52) = %loc(io$_deaccess)
0328
0329            qiocode(1,53) = %loc(io$_delete)
0330
0331            qiocode(1,54) = %loc(io$_modify)
0332
0333            qiocode(1,56) = %loc(io$_acpcontrol)
0334
0335            qiocode(1,57) = %loc(io$_mount)
0336
0337            do 10,i = 0,63
0338
0339            qiocode(0,i) = 33
0340
0341            if (qiocode(1,i) .eq. 0) then
0342
0343            qiocode(1,i) = %loc(qio_string)
0344            endif
0345
0346    10      continue
0347            endif
0348
0349            call irp$w_func (lun,emb$w_dv_func,
0350            1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0351
0352            return
0353
0354            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	255	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	2058	

ENTRY POINTS

Address	Type	Name
0-00000000		DQDISKS_QIO

VARIABLES

Address	Type	Name
AP-00000008@	I*2	EMB$W_DV_FUNC
3-00000442	CHAR	IO$_ABORT
3-000003C2	CHAR	IO$_ACPCONTROL
3-00000297	CHAR	IO$_CLEAN
3-00000385	CHAR	IO$_DEACCESS
3-0000026D	CHAR	IO$_DIAGNOSE
3-000004CB	CHAR	IO$_DSE
3-00000276	CHAR	IO$_FORMAT
3-00000014	CHAR	IO$_LOADMCODE
3-000003E2	CHAR	IO$_MOUNT
3-0000009D	CHAR	IO$_OFFSET
3-000000E0	CHAR	IO$_QSTOP
3-00000421	CHAR	IO$_READCSR
3-000002B6	CHAR	IO$_READLBLK
3-00000200	CHAR	IO$_READPRESET
3-0000033A	CHAR	IO$_READVBLK
3-00000484	CHAR	IO$_READWTHXBUF
3-0000007C	CHAR	IO$_RELEASE
3-000001B8	CHAR	IO$_REREADP
3-000002E6	CHAR	IO$_REWIND
3-000000FC	CHAR	IO$_SEARCH
3-00000231	CHAR	IO$_SENSECHAR
3-0000021D	CHAR	IO$_SETCHAR
3-00000088	CHAR	IO$_SETCLOCKP
3-000002ED	CHAR	IO$_SKIPFILE
3-00000029	CHAR	IO$_SPACEFILE
3-000003D7	CHAR	IO$_STARTDATA
3-00000037	CHAR	IO$_STARTMPROC
3-00000059	CHAR	IO$_STOP
3-0000046B	CHAR	IO$_WRITEBUFNCRC
3-000001E4	CHAR	IO$_WRITECHECKH
3-00000153	CHAR	IO$_WRITEHEAD
3-00000247	CHAR	IO$_WRITEMARK
3-0000012A	CHAR	IO$_WRITEPBLK
2-00000200	I*4	I
3-0000034D	CHAR	IO$_ACCESS
3-000004B3	CHAR	IO$_AVAILABLE
3-00000369	CHAR	IO$_CREATE
3-00000393	CHAR	IO$_DELETE
3-00000065	CHAR	IO$_DRVCLR
3-000000A9	CHAR	IO$_ERASETAPE
3-00000071	CHAR	IO$_INITIALIZE
3-000003A1	CHAR	IO$_MODIFY
3-00000000	CHAR	IO$_NOP
3-000000EB	CHAR	IO$_PACKACK
3-000003EF	CHAR	IO$_RDSTATS
3-00000169	CHAR	IO$_READHEAD
3-0000013F	CHAR	IO$_READPBLK
3-00000195	CHAR	IO$_READTRACKD
3-0000045A	CHAR	IO$_READWTHBUF
3-0000004D	CHAR	IO$_RECAL
3-000001AB	CHAR	IO$_REREADN
3-000000CA	CHAR	IO$_RETCENTER
3-000002C9	CHAR	IO$_REWINDOFF
3-00000024	CHAR	IO$_SEEK
3-00000309	CHAR	IO$_SENSEMODE
3-000003B8	CHAR	IO$_SETCLOCK
3-000002DD	CHAR	IO$_SETMODE
3-000002FA	CHAR	IO$_SKIPRECORD
3-0000010E	CHAR	IO$_SPACERECORD
3-000000B4	CHAR	IO$_STARTDATAP
3-0000020F	CHAR	IO$_STARTSPNDL
3-0000000D	CHAR	IO$_UNLOAD
3-0000011E	CHAR	IO$_WRITECHECK
3-000003FF	CHAR	IO$_WRITECSR
3-000002A2	CHAR	IO$_WRITELBLK
3-00000314	CHAR	IO$_WRITEOF
3-000001C9	CHAR	IO$_WRITERET

```
  3-0000017E  CHAR IO$_WRITETRACKD          3-00000326  CHAR IO$_WRITEVBLK
  3-00000448  CHAR IO$_WRITEWTHBUF          3-00000257  CHAR IO$_WRTTMKR
 AP-00000004@ L*1  LUN                      3-000004A1  CHAR QIO_STRING

ARRAYS

    Address  Type  Name                Bytes  Dimensions

  2-00000000  I*4  QIOCODE               512  (0:1, 0:63)

LABELS

    Address   Label

      **      10

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name

        IRP$W_FUNC         I*4  LIB$EXTZV
```

```
0001
0002


COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:DQDISKS/OBJ=OBJ$:DQDISKS MSRC$:DQDISKS

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19


COMPILATION STATISTICS

  Run Time:                10.94 seconds
  Elapsed Time:            25.02 seconds
  Page Faults:             269
  Dynamic Memory:          248 pages
```

0147 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CLASSIFY
LIS

DR750
LIS

CSTRING
LIS

DHEADS
LIS

DR780
LIS

DR11W
LIS

COMPRESS
LIS

DTAILS
LIS

DECODECC
LIS

DUMPREG
LIS

DUTUDRIVR
LIS

CALCMAP
LIS

DUP3271
LIS

CRYPTK
LIS

DQDISKS
LIS

DUP11
LIS